

METHOD AND APPARATUS FOR OVER-CLOCKING RECOVERY IN A PLL

Field of the Invention

The present invention relates to a method and/or
5 architecture for over-clocking generally and, more particularly, to
a method and/or architecture for over-clocking recovery in a phase
lock loop.

Background of the Invention

Microprocessors can be characterized by a number of
ratings. The ratings can include a nominal clock rate, an input
clock range, and an input clock slew rate. The nominal clock
rating can be conservative. A microprocessor with a conservative
nominal clock rating can have a margin of higher clock rates at
15 which the microprocessor will operate. Operating a microprocessor
above the nominal clock rating is referred to as over-clocking.

Programmable clock circuits are used by motherboard
manufacturers to enable a user to overclock the microprocessor.
However, when the over-clocking frequency exceeds the input
20 frequency range of the microprocessor, the microprocessor can lose

0325.00370
CD00060

track of the clock input and hang. In addition, if the frequency of the programmable clock circuit changes faster than the microprocessor can track (i.e., the slew rating of the microprocessor is exceeded), the microprocessor can lose track of the input clock and hang.

Referring to FIG. 1, a block diagram illustrating a microprocessor system 10 is shown. The system 10 consists of a motherboard 12, a microprocessor 14, a system block 16 that includes memory, inputs/outputs, and peripherals, and a clock circuit 18. The clock circuit 18 presents a clock signal CPUCLK to the microprocessor 14 and a clock signal SYSCLK to the system block 16. When over-clocking is desired, the frequency of the signal CPUCLK is increased above the nominal value for the microprocessor 14. If the frequency of the signal CPUCLK is set too high or changes too quickly, the microprocessor 14 will lose track and hang.

When the microprocessor 14 hangs, a recovery mechanism is required to restart the microprocessor system 10. One conventional method for over-clocking recovery is for the motherboard 12 to incorporate a separate watchdog timer or resistive-capacitive (RC) delay circuit 20. The circuit 20 resets the microprocessor after

0325.00370
CD00060

a preset amount of time if not disabled. The clock circuit 18 is reset to a default frequency for restarting the microprocessor 14. The default frequency of a conventional clock circuit 18 is set during fabrication or by jumpers 22 on the motherboard.

5

Summary of the Invention

The present invention concerns an apparatus comprising a circuit that may be configured to (i) change a frequency of one or more first signals in response to a second signal and (ii) generate a third signal in response to either the second signal or a predetermined time period expiring.

The objects, features and advantages of the present invention include providing a method and/or architecture for over-clocking recovery in a PLL that may (i) generate a reset signal when a new frequency is loaded, (ii) reset a system when the microprocessor hangs, (iii) provide a user programmable recovery frequency, (iv) require no additional reset or frequency recovery components external to the PLL circuit, (v) provide additional bus interface timing margin, (vi) automatically select an over-clocking frequency, (vii) provide relative timing adjustment between clock

0325.00370
CD00060

signals, and/or (viii) determine an optimum system operating frequency.

Brief Description of the Drawings

5 These and other objects, features and advantages of the present invention will be apparent from the following detailed description and the appended claims and drawings in which:

FIG. 1 is a block diagram illustrating a microprocessor system;

FIG. 2 is a block diagram illustrating a preferred embodiment of the present invention;

FIG. 3 is a flow diagram illustrating an example operation in accordance with a preferred embodiment of the present invention; and

15 FIG. 4 is a block diagram illustrating a preferred embodiment of the present invention.

Detailed Description of the Preferred Embodiments

Referring to FIG. 2, a block diagram of a circuit 100 is shown in accordance with a preferred embodiment of the present invention. The circuit 100 may be a programmable clock circuit for

0325.00370
CD00060

use in microprocessor systems. In one example, commands for programming the programmable clock circuit may be incorporated in the basic input/output system (BIOS) commands or program routines of a motherboard. The BIOS commands or program routines may be stored in a non-volatile computer readable medium (e.g., BIOS ROM, boot disk, etc.).

The circuit 100 may be interfaced, in one example, via an inter-integrated circuit (I²C) bus interface. However, other appropriate interfaces may be implemented to meet the design criteria of a particular application. The I²C bus may be implemented as a fully bi-directional 2-wire serial bus. The I²C bus may be compliant with the I²C specification version 1.0 (published 1992), version 2.0 (published 1998), and/or version 2.1 (published 2000) by Philips Semiconductors each of which are hereby incorporated by reference in their entirety (version 2.1 is available on the Internet at <http://www-us.semiconductors.com/i2c/>). Commands and data for programming the programmable clock circuit 100 may be sent via the I²C bus. The commands may result in the data being stored in registers, latches, programmable counters, timers, etc. of the circuit 100.

0325.00370
CD00060

The circuit 100 may be configured to present a clock signal that may allow the over-clocking of the microprocessor. The circuit 100 may be further configured to recover from a processor hang due to the over-clocking. The circuit 100 may have an input 102 that may receive a signal (e.g., EXTOSC), an input 104 that may receive a signal (e.g., SDATA), and input 106 that may receive a signal (e.g., SCLOCK), an output 108 that may present a signal (e.g., RESET), a number of outputs 110a-110n that may present a number of signals (e.g., CPUCLK, SDRAM, AGPCLK, PCICLK, APIC, etc.), and an output 112 that may present a signal (e.g., USB).

The signal EXTOSC may be implemented as a clock signal. The signal EXTOSC may generated by an external oscillator circuit 114. The signal SDATA and SCLOCK may be serial data and clock signals, respectively. The signals SDATA and SCLOCK may be implemented, in one example, in compliance with a serial communication or bus standard, for example, the inter-integrated circuit (I²C) bus specification version 1.0 (published 1992), version 2.0 (published 1998), and/or version 2.1 (published 2000) by Philips Semiconductors, each of which are hereby incorporated by reference in their entirety.

The signals CPUCLK, SDRAM, AGPCLK, PCICLK, and APIC may be implemented as system clock signals. In one example, the signals CPUCLK, SDRAM, AGPCLK, PCICLK, and APIC may be used as clock signals for a central processor unit, a synchronous dynamic random access memory, an accelerated graphics port, a peripheral component interconnect, and an advanced programmable interrupt controller, respectively. However, other numbers and uses of signals may be implemented accordingly to meet the design criteria of a particular application. For example, two or more (or all) of the components of the microprocessor system may use the same clock signal.

The signal USB may be a clock signal presented at a Universal Serial Bus (USB) port. The signal USB may be compliant with the USB specification 1.0 (published November 1996), the USB specification 1.1 (published September 1998), and/or the USB specification 2.0 (published April 27, 2000), each of which are hereby incorporated by reference in their entirety. The signal RESET may be implemented, in one example, as a system reset signal. However, more than one reset signal may be implemented accordingly to meet the design criteria of a particular application. For

0325.00370
CD00060

example, a processor reset signal may be implemented separately from a system reset signal.

The circuit 100 may comprise a circuit 120, a circuit 122, a circuit 124, a circuit 126, a circuit 128, a circuit 130, and a circuit 132. The circuit 120 may be implemented as a phase lock loop (PLL) circuit. The circuit 122 may be implemented as a programmable PLL circuit. For example, the circuit 122 may comprise a number of programmable counters, and/or a programmable VCO. The circuit 124 may be implemented as a divider network that may comprise a number of divider circuits. In one example, the divider network may comprise a number of programmable divider circuits. The circuit 126 may comprise a logic circuit that may be configured to control an amount of skew between a number of signals. The circuit 128 may be implemented as an interface circuit. In one example, the circuit 128 may be an I²C interface circuit. Alternatively, the circuit 128 may be implemented as any appropriate interface circuit for controlling and/or programming the circuit 100 in accordance with the design criteria of a particular application. The circuits 130 and 132 may be implemented as timer circuits. The circuit 130 may be used as a

0325.00370
CD00060

dividers, (ii) VCO gain, and/or (iii) any other appropriate means of varying the frequency of the signal CLK.

The circuit 124 may be implemented, in one example, as a divider network. The circuit 124 may have an input 146 that may receive a number of signals (e.g., DIVa-DIVn), and a number of outputs 148a-148n that may present a number of signals (e.g., CLK1-CLKn). In one example, the signals CLK1-CLKn may be implemented as clock signals. The circuit 124 may be configured to generate the signals CLK1-CLKn, in one example, by dividing the signal CLK in response to the signals DIVa-DIVn. Alternatively, the signals DIVa-DIVn may be used to program a number of programmable dividers that may be configured to generate one or more of the signals CLK1-CLKn in response to either the signal CLK or different ones of the signals CLK1-CLKn. Alternatively, the circuit 124 may be configured to generate the signals CLK1-CLKn in response to the signals CLK and FREQ.

Sub A1 The circuit 126 may be implemented, in one example, as a skew control logic circuit. The circuit 126 may have a number of inputs 150a-150n that may receive the signals CLK1-CLKn and an input 152 that may receive a signal (e.g., SKEW_CONTROL). The

0325.00370
CD00060

Sub A1

circuit 126 may be configured to generate a number of signals (e.g., the signals CPUCLK, SDRAM, AGPCLK, PCICLK, APIC, etc.) in response to the signal SKEW_CONTROL and the signals CLK1-CLKn. The signal SKEW_CONTROL may control an amount of skew between, for example, the signals CPUCLK, SDRAM, AGPCLK, PCICLK, and APIC. For example, the circuit 128 may be used to delay or pull-in the signal CPUCLK (or any other signal) by a preset value (e.g., within a range of 150 ps to 600 ps) relative to other signals or signal groups. In another example, the skew of one or more of the signals CPUCLK, SDRAM, AGPCLK, PCICLK, and APIC may be relative to one or more others of the signals CPUCLK, SDRAM, AGPCLK, PCICLK, and APIC. The skew may be controlled for any one or combination of the signals CPUCLK, SDRAM, AGPCLK, PCICLK, and APIC.

In one example, controlling the skew of the signals CPUCLK, SDRAM, AGPCLK, PCICLK, and APIC may provide additional timing margin for any timing critical bus interface. Controlling the skew of the signals CPUCLK, SDRAM, AGPCLK, PCICLK, and APIC may provide higher frequency over-clocking than is possible without controlling skew. The programmable skew may also provide flexibility to adjust the relative timing between the signals

0325.00370
CD00060

CPUCLK, SDRAM, AGPCLK, PCICLK, and APIC without hardware changes or a redesign of the system boards.

The signals SDATA and SCLOCK may be presented to the circuit 128. The circuit 128 may have an output 154 that may present the signals FREQ and DIVa-DIVn, an output 156 that may present the signal SKEW_CONTROL, an output 158 that may present a signal (e.g., RUNWD), and an output 162 that may present the signal RST. In one example, the signals FREQ, DIVa-DIVn, RST, and RUNWD may be generated in response to commands and/or data received using the signals SDATA and SCLOCK. The signals RST and RUNWD may be control signals. The signal RUNWD may be used, in one example, to program a watchdog interval in the circuit 130. The signal RUNWD may also be used to control the operation of the circuit 130 (e.g., starting or stopping a counting or timing operation). The signal RST may be used to program a reset time interval in the circuit 132. The signal RST may also be used to initiate a reset of the processor in response to a change of the frequency of the circuit

The circuit 130 may be implemented as a programmable counter, programmable timer, or other appropriate circuit for

0325.00370
CD00060

performing a watchdog timing function. The circuit 130 may have an input 160 that may receive the signal RUNWD and an output 166 that may present a signal (e.g., SYS_RST) to an input 168 of the circuit 132. The signal SYS_RST may be used as a system reset signal. The
5 circuit 130 may be configured to start timing (or counting) in response to the circuit 100 receiving a command to change frequency. The circuit 130 may be configured to stop timing in response to the circuit 100 receiving an indication, in one example, that the processor is executing the BIOS routines.

The circuit 130 may be configured to generate the signal SYS_RST in response to a processor hang (e.g., reaching a predetermined time or count without the circuit 100 receiving the indication that the processor is executing the BIOS routines). The watchdog timer circuit 130 may start incrementing in response to
15 the assertion of the signal RUNWD. The watchdog timer circuit 130 generally increments to a predetermined value unless the signal RUNWD is de-asserted. The predetermined value may be user programmed (via the interface circuit 128). If the signal RUNWD is de-asserted, the watchdog timer circuit 130 may return to an

0325.00370
CD00060

initial value. If the signal RUNWD remains asserted, the watchdog timer circuit 130 will generally reset the system.

The circuit 132 may be implemented, in one example, as a programmable timer or counter. However, other appropriate circuits may be implemented accordingly to meet the design criteria of a particular application. The circuit 132 may be configured to generate the signal RESET in response to either the signal RST or the signal SYS_RST. The circuit 132 may be configured to assert the signal RESET for a predetermined period of time that may be programmed in response to the signal RST. The signal RESET generated in response to the signal RST (e.g., a change in frequency) may be different (e.g., longer or shorter) from the signal RESET generated in response to the signal SYS_RST (e.g., a microprocessor hang).

Referring to FIG. 3, a flow diagram illustrating an example operation of a clock circuit implemented in accordance with a preferred embodiment of the present invention is shown. The process described below may be used to (i) recover from a processor hang due to over-clocking, and/or (ii) determine an optimum frequency for over-clocking a particular processor based system.

0325.00370
CD00060

Following power-up, a microprocessor system may begin executing instruction stored in a BIOS ROM (e.g., block 300). The instructions in the BIOS ROM may direct the microprocessor to program the circuit 100 with a new over-clocking frequency (e.g.,
5 block 302). The new over-clocking frequency may be, in one example, a predetermined (default) over-clocking frequency. Alternatively, the new over-clocking frequency may be a step in a process for determining an optimum over-clocking frequency. The reprogramming of the circuit 100 may be accomplished via an inter-
10 integrated circuit (I²C) bus. In response to the new over-clocking frequency, the circuit 100 may be configured to alter the frequency of a clock generator (e.g., a PLL, oscillator, etc.), start a watchdog timer, and present a reset signal to a pin of the microprocessor (e.g., block 304). The watchdog timer may be
15 programmed at manufacture with a predetermined (default) value or programmed via the I²C bus.

Following the reset, the microprocessor once again may begin executing instructions contained in the BIOS ROM (e.g., block 306). When the BIOS successfully starts to execute, the BIOS may
20 instruct the processor to stop the watchdog timer (e.g., block

0325.00370
CD00060

308). The processor may, in one example, send a command via the I²C bus to stop the timer circuit 130. The process may be repeated until an optimum over-clocking frequency is determined (e.g., the arrow 309). An optimum over-clocking frequency may be, in one example, the highest frequency that the processor will run at without hanging. Each time a new over-clocking frequency is used successfully to start the processor, the new frequency may be stored (e.g., in a latch, a register, a memory, etc.). When the storage medium is volatile, cycling the power may not be possible.

If a new over-clocking frequency results in a microprocessor hang, the BIOS instructions will generally not be executed and the watchdog timer will not be stopped. Once the watchdog timer reaches a maximum value (e.g., a predetermined time has passed or a maximum count is reached), a reset signal may be generated to reset the entire microprocessor system (e.g., block 310). In one example, the circuit 100 may be configured to use a predetermined (known) good frequency to clock the microprocessor following the system reset. Alternatively, the circuit 100 may be configured to use the last optimum over-clocking frequency that successfully started the system (e.g., block 312).

The circuit 100 may be configured to reset the processor with each frequency change to avoid a hang related to glitches resulting from the frequency change. The reset may be a software and/or hardware reset implemented to meet the design criteria of a particular application.

Referring to FIG. 4, a block diagram a circuit 500 illustrating a preferred embodiment of the present invention is shown. The circuit 100 may be used to over-clock a processor 502 and provide over-clocking recovery when the processor 502 hangs. The processor 502 may have an input 504 that may receive the signal CPUCLK, an input 506 that may receive the signal RESET, and an input/output 508 that may connect to communication bus 510 (e.g., an I2C bus). The circuit 100 may be programmed and controlled via the bus 510.

The present invention may provide a method and/or apparatus for over-clocking recovery in a PLL based programmable clock circuit that may (i) generate a reset signal when a new frequency is loaded, (ii) reset the system when the BIOS fails to begin executing, (iii) provide a user programmable recovery frequency, (iv) require no additional reset or frequency recovery

components external to the PLL circuit, (v) provide additional bus interface timing margin, (vi) provide high frequency over-clocking, (vii) provide relative timing adjustment between clock signals, and/or (viii) determine an optimum system operating frequency.

5 While the present invention has been described using PLL circuits, other oscillator circuits (e.g., delay lock loops, ring oscillators, etc.) may be implemented to meet the design criteria of a particular application. The present invention may be implemented using a combination of software/firmware (e.g., BIOS routines) and/or circuitry (e.g., PLLs, timers, counters, logic
10 circuits, combinatorial logic, state machines, etc.). While the present invention has been discussed in the context of particular BIOS program routines, appropriate software coding can readily be prepared by skilled programmers based on the teachings of the
15 present disclosure, as will be apparent to those skilled in the relevant art(s).

While the invention has been particularly shown and described with reference to the preferred embodiments thereof, it will be understood by those skilled in the art that various changes

19